AMENDMENT TO THE SPECIFICATION

Please amend the specification by marked up replacement paragraph(s) as follows.

On page 2, change paragraphs 1 and 2 as follows:

[01] The present application is related to U.S. Patent Application Serial No. [[_____]] 10/140,818 entitled "Method and Apparatus for Change Data Capture in a Database System" filed on [[_____]] May 9, 2002 ~~(attorney docket no. 50277-1002, client docket no. 2000-190-01)~~ by William D. Norcott et al.~~, the contents of which are hereby incorporated by reference.~~

A1

[02] The present application is related to U.S. Patent Application Serial No. [[_____]] 09/863,422 entitled "Synchronous Change Data Capture in a Relational Database ~~System~~" filed on [[_____]] May 24, 2001 ~~(attorney docket no. 50277-1006, client docket no. 2001-010-01)~~ by William D. Norcott, the contents of which are hereby incorporated by reference.

On pages 3-4, change paragraph 8 as follows:

[08] Second, there is a performance penalty in storing the timestamp for every row of new or changed data, yet performance is critical for OLTP systems. Third, while timestamps can easily identify which rows have changed ~~by~~ due to an insert or update, timestamps cannot distinguish between a newly inserted row or an old row that was updated. Furthermore, timestamps cannot identify deleted rows, because deleted rows are no longer present in the database. The lack of this kind of change information makes it difficult to properly update summaries of the OLTP data, resulting in expensive recalculations of the summary data.

A2

On page 4, change paragraph 10 as follows:

[10] Conventional systems have used triggers for synchronous change data capture, either by using the CREATE TRIGGER statement or by using an internal mechanism with equivalent functionality. A trigger is an object that specifies a series of actions to be automatically performed when a specific event occurs, and, according to industry standards, the events that cause triggers to be activated (or "fired") are ~~DML~~ Data Manipulation Language (DML) statements. For synchronous change data capture, triggers have been designed to fire when a row

A3

A3

of a database table is updated, inserted, or deleted. Each of these modifications is associated with its own system change number (SCN), which is recorded by the trigger.

On page 5, change paragraphs 14-15 as follows:

---

[14] Accordingly, one aspect of the present invention involves a method and software for change data capture, in which change data is extracted from a recovery ~~logand~~ log and stored in a database object such as a change table. The change data stored in the database object indicates what modification has been performed to a source object on the OLTP system. In various embodiments of the present invention, a database statement may be generated and executed to extract and load the change data. The recovery log itself may be shipped from an OLTP system to a staging system.

A4

[15] Still other aspects, features, and advantages of the present invention are readily apparent from the following detailed description, simply by illustrating a number of particular embodiments and implementations, including the best mode contemplated for carrying out the present invention. The present invention is also capable of other and different embodiments, and its several details can be modified in various obvious respects, all without departing from the spirit and scope of the present invention. Accordingly, the ~~drawing~~ drawings and description are to be regarded as illustrative in nature, and not as restrictive.

On pages 8-9, change paragraphs 26-29 as follows:

---

[26] In the synchronous extraction mechanism, triggers 115 are employed to capture each change to the OLTP database 113 when the changes are made and transport the changes to the staging system 120. At the staging system 120, these changes are then integrated and loaded into change tables (not shown) of the analysis database 125 by a publisher process 127. The synchronous extraction mechanism is described in greater detail in the commonly assigned, co-pending U.S. Patent Application Serial No. [[_____]] 09/863,422 entitled "Synchronous Change Data Capture in a Relational Database ~~System~~" filed on [[_____]] May 24, 2001 ~~(attorney docket no. 50277-1006, client docket no. 2001-010-01)~~ by William D. Norcott, the contents of which are hereby incorporated by reference.

A5

[27] For the asynchronous extraction mechanism, which is described in greater detail herein below, a log shipper 119 periodically copies recovery logs 117 that are produced by the OLTP database 113 in the normal course of operation. The recovery logs 117 contain all the changes that have been applied to the OLTP database 113 and are used for backing up the data in the OLTP database 113 and restoring the data in case of a system crash. The log shipper 119 copies the recovery logs 117 to an area of the staging system 120 called a change source 131, which can be implemented as an operating system directory. The publisher 127 interacts with a log viewer process 129 to obtain the change data from the shipped recovery logs in the change source 129 without having to be aware of the internal implementation details of the recovery logs. The publisher 127 then loads the change data obtained via the log viewer process 129 into the change tables in the analysis database 125 for use by the subscriber applications 121, 123. ~~The interaction of the subscriber applications 121, 123 with the change tables is described in the commonly assigned, co-pending U.S. Patent Application Serial No. _____ entitled "Method and Apparatus for Change Data Capture" filed on _____ (attorney docket no. 50277-1002, client docket no. 2000-190-01) by William D. Norcott et al., the contents of which are hereby incorporated by reference.~~

OBJECTS FOR MANAGING CHANGE DATA

[28] In accordance with one aspect of the present invention, the change data extracted from the OLTP database 113 is maintained not in a flat file but in one or more database objects, referred to herein as "change tables" under control of a database management system, e.g. analysis database ~~123~~ 125. Because the database management system provides such features as crash recovery for security and indexing for performance, use of database objects to hold the change data advantageously attains these beneficial features, without additional programming as compared to use of flat files.

[29] Referring to FIG. 2 by way of example, each source table or database object on the OLTP database 113 that is subject to change data capture is associated with a corresponding change table 211, 213, 221, 223, 225 in the analysis database ~~123~~ 125. For transactional consistency, change tables 211, 213, 221, 223, 225 are grouped into sets of one or more "change sets " 210, 220 such that the publisher ~~125~~ 127 ensures that all new change data added to the change tables in the same change set (e.g. changes tables 211, 213 of change set 210) are added at the same

time, e.g. the modifications to these changes tables are performed in the same transaction and committed. In the example depicted in FIG. 2, there are two change sets, change set 210 and change set 220. Change set 210 comprises change table 211 and change table 213, which correspond to respective tables (not shown) on the OLTP database 113. Likewise, change set 220 comprises change table 221, change table 223, and change table 225, which also correspond to respective tables (not shown) on the OLTP database 113. The information that defines the structure of the change sets 210, 220 and change tables 211, 213, 221, 223, 225 is maintained in system metadata 230.

A5

On page 10, change paragraph 32 as follows:

[32] The SCN 233 column holds the System Change Number of the commit for the transaction on the OLTP database 113 that gave rise to the change data. A system change number is a monotonically increasing number that identifies every operation performed on a database system, e.g. update, insert, delete, and commit, that can be used to order the operations performed in the database system. The present invention is not limited to any particular ~~implement~~ implementation of system change numbers, and the concepts disclosed herein may be profitably employed with timestamps, incrementing serial numbers, and the like.

A6

On pages 11-12, change paragraphs 36-38 as follows:

[36] Although not depicted in FIG. 2, additional control columns may be provided to facilitate the implementation of embodiments of the present invention. For example, a bit mask of the updated columns can be used to identify quickly which columns have changed. As another example, the name of a user who ~~cause~~ causes the operation can be recorded in a control column. The row identifier of the affected row in the source table can also be included in a control column.

A7

[37] The subscriber applications 121, 123 of FIG. 1, however, need not see all the contents of the change tables. In particular, the range of rows that the subscriber ~~application~~ applications 121, 123 ~~sees~~ see in the respective subscriber ~~view~~ views is restricted and carefully controlled, as explained in detail below, so that change data is not lost nor double counted for subscriber

applications 121, 123. In the example of FIG. 2, two subscriber views 241, 243 are depicted although any number of subscriber views may be created during the operation of an embodiment of the present invention. Use of the subscriber views 241, 243 beneficially insulates the respective subscriber applications 121, 123 from the implementation details of the change table 225. Unlike some prior art approaches, it is not necessary to add any control columns or information to the source tables themselves on the OLTP database 113; the provision of control columns for the change tables on the analysis ~~120~~ database 125 suffices. Consequently, this feature allows change data capture to be performed without changing the schema of the OLTP database 113, which is desirable for many turn-key OLTP databases.

ASYNCHRONOUS CHANGE DATA CAPTURE

A7

[38] Referring to FIG. 3, at step ~~310~~ 301, the recovery logs 117 are shipped from the source system 110 to a location on the staging system 120 called a change source 131, which can be an operating system directory. Also known as a "redo log," a recovery log 117 is produced by the OLTP database system 113 in the normal course of operation to allow users to undo transactions (e.g. in a transaction rollback) and to provide for recovery after a system crash. Recovery logs 117 thus provide a way to cancel or to abort a transaction before the transaction is committed, and recovery logs 117 are a standard feature of relational database management systems. One way to implement the log shipper 119 is described in the commonly assigned, co-pending U.S. Patent Application Serial No. [[_____]] 09/852,008 entitled "Disaster Recovery with Bounded Data Loss" filed on May 10, 2001 ~~(attorney docket no. 50277-1003, client docket no. 2000-207-01)~~ by Mahesh Girkar et al., the contents of which are hereby incorporated by reference.

On page 13, change paragraph 41 as follows:

A8

[41] At step 305, each change table is processed in a change set, which is a logical group of related change tables that are to be kept transactionally consistent. Accordingly, a transaction to update the change tables is begun for all the change tables in the change set and is committed at the end of this process to ensure that all the changes to related change ~~table~~ tables will become visible to the subscriber applications 121, 123 at the same time.

On page 14, change paragraph 45 as follows:

[45] The slot "*source_columns*" is a list of the columns in the change table. In this example, if the SCOTT.EMP source table contains source columns EMPNO, ENAME, HIREDATE, and DEPTNO, then the "*source_columns*" slot would be filled with "EMPNO, ENAME, HIREDATE, DEPTNO". The string for this list may be built by iterating over the information stored in system metadata 230.

A9

On page 15, change paragraph 49 as follows:

[49] At step 309, the generated SQL statement is executed, thereby fetching the change data from the shipped recovery logs 117 via the log viewer 129 and update the corresponding change tables. The SQL "INSERT ... SELECT" statement can be optimized by the analysis database 125 when its native language is SQL. As a result, the analysis database 125 is able to take advantage of the indexing, partitioning, and parallel features of the analysis database 125. Furthermore, use of standard SQL for this operation benefits from the data integrity and transactional capability that is the hallmark of a modern relational database management system. For example, a plurality of the SQL "INSERT ... SELECT" statements can be made part of a single transaction per change set, ~~to~~ so that the entire data loading operation executes as a single transaction. This approach addresses data integrity or recovery problems that characterize the conventional flat file approach that would arrive, for example, if columns EMPNO, ENAME, and HIREDATE successfully loaded, but column DEPTNO contained an error, impacting transactional consistency.

A10

On pages 17-18, change paragraphs 55-57 as follows:

[55] The computer system 400 can send messages and receive data, including program code, through the network(s), network link 419, and communication interface 417. In the Internet example, a server (not shown) might transmit requested code belonging to an application program for implementing an embodiment of the present invention through the network 425, local network 421 and communication interface 417. The processor ~~404~~ 403 may execute the

A11

transmitted code while being received and/or store the code in storage device ~~49~~ 409, or other non-volatile storage for later execution. In this manner, computer system 400 may obtain application code in the form of a carrier wave.

[56] The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to the processor ~~404~~ 403 for execution. Such a medium may take many forms, including but not limited to non-volatile media, volatile media, and transmission media. Non-volatile media include, for example, optical or magnetic disks, such as storage device 409. Volatile media include dynamic memory, such as main memory 405. Transmission media include coaxial cables, copper wire and fiber optics, including the wires that comprise bus 401. Transmission media can also take the form of acoustic, optical, or electromagnetic waves, such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, CDRW, DVD, any other optical medium, punch cards, paper tape, optical mark sheets, any other physical medium with patterns of holes or other optically recognizable indicia, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave, or any other medium from which a computer can read.

A11

[57] Various forms of computer-readable media may be involved in providing instructions to a processor for execution. For example, the instructions for carrying out at least part of the present invention may initially be borne on a magnetic disk of a remote computer. In such a scenario, the remote computer loads the instructions into main memory and sends the instructions over a telephone line using a modem. A modem of a local computer system receives the data on the telephone line and uses an infrared transmitter to convert the data to an infrared signal and transmit the infrared signal to a portable computing device, such as a personal digital ~~assistance~~ assistant (PDA) and a laptop. An infrared detector on the portable computing device receives the information and instructions borne by the infrared signal and places the data on a bus. The bus conveys the data to main memory, from which a processor retrieves and executes the instructions. The instructions received by main memory may optionally be stored on a storage device either before or after execution by processor.